4.1 Programming and Usability (Ken 3-4 pages)

4.1.1 Programming Models

State of the Art:
Although many HPC workflows are hand constructed through shell scripts, batch scheduling, and human intervention, there exist programmable tools such as Swift [Wilde2011], Tigres [Ramakrishnan2014], and many others [Altintas2004, Barga2008, Bui2010, Churches2006, Deelman2005, Goecks2010, Oinn2006] to better manage complicated workflows. The Open Provenance Model (OPM) [Moreau2011] is an open standard specification of a provenance data model with multiple existing compliant implementations [Cuevas-Vicenttín2012, Garijo2012, Lim2010]. Programming models for cloud, web service, and other big data applications are abundant. The programming model for MapReduce [Dean2004] is probably the most well known, but many others exist [Fox2014, Jha2014, Qiu2014]. Many of these models may be leveraged for use in high performance computing.

[Altintas2004] I. Altintas, C. Berkley, E. Jaeger, M. Jones, B. Ludscher, and S. Mock. Kepler: “An extensible system for design and execution of scientific workflows.” In *Proceedings of the 16th International Conference on Scientific and Statistical Database Management*, pages 423–424, 2004.

[Barga2008] R. Barga, J. Jackson, N. Araujo, D. Guo, N. Gautam, and Y. Simmhan. “The trident scientific workflow workbench.” In *IEEE Fourth International Conference on eScience*, pages 317 –318, dec. 2008.

[Bui2010] P. Bui, L. Yu, and D. Thain. “Weaver: integrating distributed computing abstractions into scientific workflows using python.” In *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing, HPDC ’10*, pages 636–643, 2010. DOI: 10.1145/1851476.1851570.

[Churches2006] D. Churches, G. Gombas, A. Harrison, J. Maassen, C. Robinson, M. Shields, I. Taylor, and I. Wang. “Programming Scientific and Distributed Workflow with Triana Services.” *Concurrency and Computation: Practice and Experience (Special Issue: Workflow in Grid Systems)*, 18(10):1021–1037, 2006.

[Cuevas-Vicenttín2012] Cuevas-Vicenttín, V.; Dey, S.; Wang, M. Y.; Song, T.; Ludäscher, B. “Modeling and querying scientific workflow provenance in the D-OPM.” In *High Performance Computing, Networking, Storage and Analysis (SCC)*, 2012 SC Companion Technology, pp. 119-128, 2012. DOI: 10.1109/SC.Companion.2012.27.

[Dean2004] J. Dean and S. Ghemawat. “MapReduce: Simplified Data Processing on Large Clusters.” In *OSDI*, pages 137–150, 2004.

[Deelman2005] E. Deelman, G. Singh, M. Su, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, G. Berriman, J. Good, et al. “Pegasus: A framework for mapping

complex scientific workflows onto distributed systems." *Scientific Programming*, 13(3):219–237, 2005.

[Fox2014] Geoffrey Fox, Judy Qiu, and Shantenu Jha. "High Performance High Functionality Big Data Software Stack." In *Big Data and Extreme-scale Computing (BDEC)*. 2014.

[Garijo2012] Garijo, D.; Gil, Y. "Towards Open Publication of Reusable Scientific Workflows: Abstractions, Standards and Linked Data." Internal Project Report. 2012. Accessed on March 23, 2015 at: http://www.isi.edu/~gil/papers/garijo-gil-opmw12.pdf.

[Goecks2010] J. Goecks, A. Nekrutenko, J. Taylor, and The Galaxy Team. "Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences." *Genome Biology*, 11(8):R86, 2010.

[Jha2014] Shantenu Jha, Judy Qiu, Andre Luckow, Pradeep Mantha, and Geoffrey C. Fox. "A Tale of Two Data-Intensive Approaches: Applications, Architectures and Infrastructure." In *3rd International IEEE Congress on Big Data*. June 27- July 2, 2014.

[Lim2010] Lim, C.; Lu, S.; Chebotko, A.; Fotouhi, F. "Storing, reasoning, and querying OPM-compliant scientific workflow provenance using relational databases." *Fut. Gener. Comput. Syst.* 27(6):781-789, 2010. DOI: 10.1016/j.future.2010.10.013.

[Moreau2011] Moreau, L.; Clifford, B.; Freire, J.; Futrelle, J.; Gil, Y.; Groth, P.; Kwasnikowska, N.; Miles, S.; Missier, P.; Myers, J.; Plale, B.; Simmhan, Y.; Stephan, E.; Van den Bussche, J. "The Open Provenance Model core specification (v1.1)." *Fut. Gener. Comput. Syst.* 27(6):743-756, 2011. DOI: 10.1016/j.future.2010.07.005.

[Oinn2006] T. Oinn et al. "Taverna: lessons in creating a workflow environment for the life sciences." *Concurrency and Computation: Practice and Experience*, 18(10):1067–1100, 2006.

[Qiu2014] Judy Qiu, Shantenu Jha, Andre Luckow, and Geoffrey C.Fox. "Towards HPC-ABDS: An Initial High-Performance Big Data Stack." In *Building Robust Big Data Ecosystem ISO/IEC JTC 1 Study Group on Big Data*. March 18-21, 2014.

[Ramakrishnan2014] Ramakrishnan, Poon, Hendrix, Gunter, Pastorello, and Agarwal. "Experiences with User-Centered Design for the Tigres Workflow API." In *IEEE 10th International Conference on e-Science*, pages 290-297, October 2014. DOI: 10.1109/eScience.2014.56.

[Wilde2011] M. Wilde, M. Hategan, J. M. Wozniak, B. Clifford, D. S. Katz, and I. Foster. "Swift: A language for distributed parallel scripting." *Parallel Computing* 37(9), 2011.

Challenges:

As leadership class machines and the workflows applied to them increase in complexity, the horizon between workflow and programming model becomes blurred. The workflow system's representation of a mix of coarse data- and task-parallelism mirrors the finer-grained task-parallel computations that are predicted for increasing parallelism on extreme-scale systems beyond the common data-parallelism in applications. It is unclear where the line between responsibilities of workflow and programming model should lie or indeed if there should be a separation at all. A combined hierarchical representation of tasks characterizing the spectrum of parallelism from coarse-grained jobs to fine-grained processing threads may be most effective.

In addition to better understanding the relationship between workflows and programming models, several other challenges exist.

- Mapping: The workflow exists in its own abstract model. This workflow abstraction must be mapped to the physical compute, analysis, and storage resources while taking into account an accurate model of their relative costs. These decisions may define how workflows are composed. [I admit that I'm not entirely sure what the notes meant when it says "compose workflows." Perhaps we can word this better.]
- Provenance: The capture of provenance is critical for future analysis and reproducibility even for teams that do not otherwise rely on workflow systems. It is important to capture all relevant information including information only available after workflow execution (such as user information regarding quality of solution).
- Pragmatics: Just as science teams exhibit different workflow patterns, their modes of interaction with systems, the data they use and generate, and their rate of adoption of workflow related tools will vary.

R&D Needed:

As both workflows and programming models address high concurrency, dynamic application execution, dynamic resource availability, architectural diversity and heterogeneity, and new forms of in-system storage in extreme-scale architectures, research must manage the gaps in increased complexity.

- Horizon and Coordination: As the resource allocation responsibilities of workflows and programming models overlap, integration and coordination becomes fruitful and perhaps necessary. Programming models and system introspection could reflect knowledge up to the workflow manager or even to the programmer.
- Mapping: Domain-specific workflow systems could be used to exploit proposed programming systems that assemble applications composed of different algorithms or implementations based on execution context. This would simplify the mapping of workflows to the diversity of resources as compared to a more general workflow system.
- Provenance: For workflows that are dynamically executed as a result of system resource allocation or steered by the user to change the computation, we need to capture run-time decisions that affect execution even if it is not possible to replicate the exact execution.

- Pragmatics: Workflow systems must provide productive interfaces, and workflow tools should be decomposed in such a way that science teams can iteratively adopt components into their existing work practices.

4.1.2 Design Patterns

State of the Art:
The basic mental model for a workflow is a directed acyclic graph (DAG) representing the tasks to perform and the dependencies between the tasks. Workflow management tools such as Swift [Wilde2011] and Tigres [Ramakrishnan2014] center their API around building DAGs and internally manage parallel execution and dependencies with them. Other tools like AVS [Upson1989], SCIRun [Parker1995], and VisTrails [Bavoil2005] allow users to build and view workflow DAGs visually with a graphical representation and user interface.

Many scientists build workflows by example, which is an informal building and use of design patterns; they will iteratively construct one workflow using a previous one as a template. This incremental use simplifies the process of building and running workflows. This iterative addition to workflows and their capabilities shortens the time to user payback and can also be integrated in software design to accelerate workflow tool development [Maheshwari2013].

Some recent tools use pattern-like structures as part of the creation and execution of workflows. For example, Tigres has a collection of templates that can be applied when building workflow structures [Balderrama2014]. VisTrails can collect the provenance of many previously built workflows to find common patterns that can automatically assist users in other endeavors [Silva2007].

Challenges:
Many workflows, particularly those from the same group, are similar. These can potentially be defined as patterns with data, error control, and reproducibility. They can be characterized by their resource access — network traffic, high throughput, etc — to take the best advantage of resources while being portable. Science teams exhibit a variety of different workflow patterns, particularly across communities. Understanding these patterns can aid in the design of workflows and workflow systems.

However, it is unclear that any single workflow management system would be effective for supporting this variety of patterns. There are different types of scientific workflow needs that can change depending on the domain or the mode of operation such as production versus real-time versus exploratory.

Although a DAG is the fundamental model used to describe workflows and their patterns, they cannot capture interdependencies among tasks, which results in cycles in the dependency graph. Such interdependencies are common in, for example, multiphysics codes where independent tasks rely on the models and iterative computation of one another to converge to the appropriate solution. Such cycles can sometimes be handled by conditionals or dynamic scripting in the workflow tool, but better descriptions of the workflow could be made.

R&D Needed:
It is important to understand and classify various workflow and workflow needs through user research. Identifying common patterns, akin to design patterns in software engineering [Gamma1995], for next-generation in-situ and distributed workflows is needed to address programmability and usability concerns.

Workflows need to correctly and more formally handle task dependency loops. Part of this requires workflows to understand and manage time and data that changes over time, which has been demonstrated in the similar VTK dataflow network [Biddiscombe2007].

[Balderrama2014] Javier Rojas Balderrama, Matthieu Simonin, Lavanya Ramakrishnan, Valerie Hendrix, Christine Morin, Deborah Agarwal, and Cedric Tedeschi. “Combining Workflow Templates with a Shared Space-based Execution Model.” In *Proceedings of the 9th Workshop on Workflows in Support of Large-Scale Science (WORKS ‘14)*, 2014. DOI: 10.1109/WORKS.2014.14

[Bavoil2005] Louis Bavoil and Steven P. Callahan and Patricia J. Crossno and Juliana Freire and Carlos E. Scheidegger and Claudio T. Silva and Huy T. Vo. “VisTrails: Enabling Interactive Multiple-View Visualizations.” In *Proceedings of IEEE Visualization*, October 2005.

[Biddiscombe2007] John Biddiscombe, Berk Geveci, Ken Martin, Kenneth Moreland, and David Thompson. “Time Dependent Processing in a Parallel Pipeline Architecture.” *IEEE Transactions on Visualization and Computer Graphics*, 13(6), November/December 2007. DOI: 10.1109/TVCG.2007.70600.

[Gamma1995] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison Wesley, 1995.

[Maheshwari2013] Ketan Maheshwari, David Kelly, Scott J. Krieder, Justin M. Wozniak, Daniel S. Katz, Zhi-Gang Mei, and Mainak Mookherjee. “Reusability in science: From initial user engagement to dissemination of results.” In *Proc. Workshop on Sustainable Software for Science: Practice and Experiences at SC*, 2013.

[Parker1995] Steven G. Parker and Christopher R. Johnson. “SCIRun: A Scientific Programming Environment for Computational Steering.” In *Proceedings ACM/IEEE Conference on Supercomputing*, 1995.

[Silva2007] Cláudio T. Silva, Juliana Freire, and Steven P. Callahan. “Provenance for Visualizations: Reproducibility and Beyond.” *Computing in Science & Engineering*, 9(5), September/October 2007. DOI: 10.1109/MCSE.2007.106.

[Upson1989] Craig Upson, Thomas Faulhaber Jr., David Kamins, David Laidlaw, David Schlegel, Jeffrey Vroom, Robert Gurwitz, and Andries van Dam. “The Application Visualization System: A Computational Environment for Scientific Visualization.” *IEEE Computer Graphics and Applications*, 9(4), July 1989.

[Also using some references listed in Section 4.1.1.]

4.1.3 User Interface

State of the Art:

Most workflow management tools use a scripting language to define tasks and dependencies and to manage execution [Altintas2004, Barga2008, Bui2010, Churches2006, Deelman2005, Goecks2010, Oinn2006, Ramakrishnan2014, Wilde2011]. Within the scripting language is an API that scripts use to define and execute a workflow. There also exist examples of workflow building tools that use an interface that provides a graphical representation of the workflow that is directly manipulatable [Parker1995, Bavoil2005]. Such interfaces provide a tradeoff between simplicity and expressibility.

Challenges:

Today, workflows are modeled in many different ways including scripts and application/programming models. The boundaries and interactions between the representation of workflow constructs and application interaction such as loops and parameter convergence/divergence are not well understood.

There is a constant tension between generalizing and specializing the workflow user interface. A generalized interface provides a greater amount of expression that can address more domains and more anomalous cases whereas a specialized interface tends to be easier to use and could provide more opportunities to optimize the workflow process.

Human-in-the-loop occurs in many different types of workflows including exploratory, failure recovery in production pipelines. The real time status of the workflow needs to be accessible to users. Many simulations require human monitoring for erroneous conditions such as the entangling of a mesh. Real time status is also important for observational and experimental data. For example, a human may be required to determine if missing telescope data is a result of a cloudy night or failures in the hardware or software. Beamline experiments can benefit from real-time feedback to improve experimental decisions [I think this is the same as the experiment talked about by Richard Carlson (slide 5). I don't know the reference but it would be good to put here.]. There is limited support to enable seamless integration of the human in today's workflows and automatically track the provenance from such activities.

R&D Needed:

It is unclear what is the appropriate level of abstraction for workflows. This level of abstraction may not be uniform for all use cases. Different domains may require different abstractions. It could also be beneficial to have different levels of abstractions for early and advanced users. Given a level of abstraction, what hints can a user provide to better map the workflow to the available resources?

Providing the interface for human-in-the-loop workflows is critical, but there is as yet no known way to properly capture the provenance of human interaction. To preserve repeatability, we must find ways to capture when a human makes changes, what changes were made, and the reason the change was made.

4.1.4 Task Communication

State of the Art:
Many current and past workflows have used files to communicate data between programs. For such an interface to work, the programs must understand each other's file format. Thus, many HPC file formats are "self describing" in that arrays are organized using names, attributes, and hierarchies [Lofstead2008, Prabhat2014]. Similar organization is present in array databases [Stonebraker2011] and NoSQL databases [Cattell2010]. This organization, however, has little meaning without an agreement on the semantics. Thus, conventions [Eaton2011] and schemas [Clarke2007, Shasharina2010, Tchoua2013] are often applied.

Other work has focused on providing a unified interface to data that can come from a variety of storage implementation. For example, ADIOS can support numerous I/O back ends and switch between them at run time [Lofstead2008]. Tools like Google Dremmel [Melnik2010] and Apache Drill provide a unified interface to multiple data backends.

Challenges:
Today's workflows and workflow systems have limited support for considering data sources, storage needs and data models. As workflows integrate ever varying tasks it becomes more challenging to communicate data between software that uses different data models. Systems that attempt to provide a unified data model have often been unsuccessful. Many file and database systems provide successful mechanisms for declaring data format, but these mechanisms need to be expanded and applied to direct task-to-task communication within a workflow.

R&D Needed:
We need to develop appropriate infrastructure that allows for seamless integration of various data sources including streaming, data management across the memory-storage hierarchy of next generation systems and data semantics/model in user workflows. More needs to be understood how data can be communicated among tasks that are developed independently and have different data models. Data can be thought of as stored, streamed in and coming from multiple sources. The mechanism for connecting these sources as well as the ability to identify, convert, and verify data models must be well established. This communication will need to take place over a variety of levels in a deep memory hierarchy.

[Cattell2010] Rick Cattell. "Scalable SQL and NoSQL Data Stores." *ACM SIGMOD Record*, 39(4), December 2010. DOI: 10.1145/1978915.1978919.

[Clarke2007] Jerry A. Clarke and Eric R. Mark. "Enhancements to the eXtensible Data Model and Format (XDMF)." In *DoD High Performance Computing Modernization Program Users Group Conference*, June 2007. DOI: 10.1109/HPCMP-UGC.2007.30.

[Eaton2011] Brian Eaton, et al. *NetCDF Climate and Forecast (CF) Metadata Conventions*, version 1.6. December 2011. http://cfconventions.org/Data/cf-conventions/cf-conventions-1.6/build/cf-conventions.pdf

[Lofstead2008] Jay F. Lofstead, Scott Klasky, Karsten Schwan, Norbert Podhorszki, and Chen Jin. “Flexible IO and Integration for Scientific Codes Through The Adaptable IO System (ADIOS).” In *Proceedings of the 6th international workshop on Challenges of large applications in distributed environments (CLADE ‘08)*, 2008. DOI: 10.1145/1383529.1383533.

[Melnik2010] Sergey Melnik, Andrey Gubarev, Jing Jing Long, Geoffrey Romer, Shiva Shivakumar, Matt Tolton, and Theo Vassilakis. “Dremel: Interactive Analysis of Web-Scale Datasets.” In *Proceedings of the VLDB Endowment*, 3(1-2), September 2010. DOI: 10.14778/1920841.1920886.

[Prabhat2014] Prabhat and Quincey Koziol. *High Performance Parallel I/O*. CRC Press, 2014.

[Shasharina2010] S. Shasharina, J. Cary, M. Durant, D. Alexander, S. Veitzer, and S. Kruger. “Vizschema - A Unified Visualization of Computational Accelerator Physics Data.” In *Proceedings of IPAC’10*, 2010.

[Stonebraker2011] Michael Stonebraker, Paul Brown, Alex Poliakov, and Suchi Raman. “The Architecture of SciDB.” In *Scientific and Statistical Database Management*, 2011. DOI: 10.1007/978-3-642-22351-8_1.

[Tchoua2013] Roselyne Tchoua, Jong Choi, Scott Klasky, Qing Liu, Jeremy Logan, Kenneth Moreland, Jingqing Mu, Manish Parashar, Norbert Podhorszki, David Pugmire, and Matthew Wolf. “ADIOS Visualization Schema: A First Step Towards Improving Interdisciplinary Collaboration in High Performance Computing.” In *IEEE International Conference on eScience*. October 2013. DOI: 10.1109/eScience.2013.24.

4.1.5 Portability

Many analysis codes are run sometimes as in situ processes, meaning that all tasks run within a local supercomputer’s resources, and are sometimes run as distributed area processes where tasks are coordinated across multiple independent systems, which might be physically distant from each other. We do not wish to implement these workflows twice. Therefore, their interface needs to be designed to work in either in situ or distributed area modes. For example, the same code may need to point to data in memory, read data from a file, output data to memory or a file, run in serial or parallel, compute a small-scale or large-scale job, process data in-core or out-of-core, and be built as a library or as an executable. All of this ought to be uniform so data and control can seamlessly flow between in situ environments and data analysis environments.

State of the Art:

There are no widely adopted general-purpose workflow tools available that work seamlessly across both in situ environments and distributed area environments. However, there are many examples of specific applications designed to work in both domains. CyberShake, a seismic hazard model from the Southern California Earthquake Center [Graves2011], combines components that use high-performance computing and high-throughput computing. The Advanced Photon Source coordinates high-performance computation with detector hardware and other processing systems [Khan2013]. Likewise, the National Synchrotron Light Source II has initial processing in situ with data collection with the results sent to distributed users. The HACC cosmology simulation can interface its high-performance computation with other analysis on other systems through the CosmoTools analysis framework [Habib2014]. KBase, the DOE systems biology knowledgebase, contains in situ modeling and reconstruction tools as well as offloading to cloud distributed area systems [Benedict2014].

[Benedict2014] Matthew N. Benedict, Michael B. Mundy, Christopher S. Henry, Nicholas Chia, and Nathan D. Price. “Likelihood-Based Gene Annotations for Gap Filling and Quality Assessment in Genome-Scale Metabolic Models.” *PLOS*, October 2014. DOI: 10.1371/journal.pcbi.1003882.

[Graves2011] Robert Graves, Thomas H. Jordan, Scott Callaghan, Ewa Deelman, Edward Field, Gideon Juve, Carl Kesselman, Philip Maechling, Gaurang Mehta, Kevin Milner, David Okaya, Patrick Small, and Karan Vahi. “CyberShake: A Physics-Based Seismic Hazard Model for Southern California.” *Pure and Applied Geophysics*, 168(3-4), March 2011. DOI: 10.1007/s00024-010-0161-6.

[Habib2014] Salman Habib, Adrian Pope, Hal Finkel, Nicholas Frontiere, Katrin Heitmann, David Daniel, Patricia Fasel, Vitali Morozov, George Zagaris, Tom Peterka, Venkatram Vishwanath, Zarija Lukić, Saba Sehrish, and Wei-keng Liao. “HACC: Simulating Sky Surveys on State-of-the-Art Supercomputing Architectures.” [I couldn’t figure out the status of this publication. Since Tom is a coauthor, maybe he knows.]

[Kahn2013] F. Khan, J.P. Hammonds, S. Narayanan, A. Sandy, and N. Schwarz. “Effective End-to-end Management of Data Acquisition and Analysis for X-ray Photon Correlation Spectroscopy.” In *Proceedings of ICALEPCS 2013*, October 2013.

Challenges:
To address workflows both across in situ and distributed area boundaries and across applications, we need to find a common language for building workflow tools. This may be a job for the emerging field of “workflow science,” which itself is highly related to (and perhaps a subset of) data science.

We expect steering and human interaction to become more important in the future. Better tools are needed to express and enable dynamic control while not interfering with the run. Many problems require a human in the loop as the most complex decisions cannot be programmed. Furthermore, scientists innately like to know how

their run is going and perhaps need to decide what to do in case of failures. It is not clear how this projects to in situ versus distributed area space.

In situ workflows often make different assumptions about the abilities of users than distributed area workflows: In situ workflows usually assume more computationally sophisticated users who are comfortable with programming and scripting whereas distributed area workflows usually assume less sophisticated users who may need a more user-friendly interface. There is not necessarily good technical justification for these assumptions though they may be related to the history of the domains that respectively use in situ workflows and distributed area workflows.

Time plays a significant and different role for in situ and distributed area connections. In situ is generally real-time in that data is transient and analysis must keep up with the simulation or throttle the speed of the job. In contrast distributed area generally has looser scheduling because the lifespan of the data is longer. There also may be a difference in how results of in situ and distributed area workflows are stored and used: Distributed area scientists often collect data for long-term sharing whereas in situ data is often shared within smaller groups that need the data for less time. However, there are also counterexamples, such as climate, material science, and cosmology, where in situ workflows produce data that is intended to be used by large communities over a long time.

Portability is difficult on complex, heterogeneous systems. Part of the solution falls to other areas of research (such as programming models), but workflows can also help. Workflows can help match tasks to the architecture best suited to run them; this is more common in distributed area workflows, but it is still an area of research for in situ. Containers, with workflows operating above the container level, are a possible solution for some distributed area and in situ workflow issues.

R&D Needed:
How can we build workflow systems and common application components that operate with good performance in both distributed areas and in situ?

What is the role of containers and other virtualization technologies in workflows? Virtualization can have a profound effect both for workflow components (tasks) and for the workflow systems themselves.

Security for in situ workflows often relies on the access controls of the system whereas distributed area workflows must be more cognizant of security since they run across different systems in different security domains. Often the security policies between in situ, distributed area, and the compute facilities are in conflict with each other. How can security be unified across all these elements to allow application workflows to best be developed, deployed, and executed?

Can workflows be leveraged to manage deep memory hierarchies? Given an appropriate decomposition of the problem, as workflows orchestrate tasks they can manage the movement of data up and down this memory hierarchy. For example, workflows could potentially manage movement of data between out-of-core and in-

core, between NVRAM and main memory, and between main RAM and high bandwidth RAM.

As high-performance computing and workflows become more complex, the interaction between human and system becomes more important. For large scale applications it is not feasible to continually monitor tasks and restart jobs when problems occur or steering is necessary. Rather, analysis tasks need a better communication path to analyst and interaction to modify or correct behavior must be possible. The human interaction can become even more complex as we mix in situ and distributed area coupling in the same workflow.

4.1.6 Ties to other Sections

Programming and usability crosscuts many workflow topics. Requirements for provenance (discussed at length in Section 4.3) can have a profound effect on the programmability of the system. Models and proxy applications (Section 4.4.1) are an important tool in the design of programming models and design patterns as they can help predict the behavior of more complicated systems. Data models and data management (Section 4.2.2) are a critical component of the task communication. Many aspects of the hardware (Section 3.2.1) for the basis for the appropriate abstractions in programming models and design patterns. Finally, well understood workflow science (Section 4.5) is required to build workflow systems that are both usable and effective.

Dropped on the floor

Can we prove the correctness of workflows?

Real-time workflows are producing data and need to understand it immediately. Recovery should allow scaling to catchup. Previous data is used for future analyses. Analogies with financial markets.

Production workflows are not always real-time. Robust and reliable – serves analyses needs of many people. Are on a timeline of publications. Do require follow-up at the same time? Processed multiple times -> improving algorithms. Human in the loop is prevalent in the loops here.

Workflow interaction with system resources for optimization, auto-tuning and portability are relatively challenging.

Experimentation, modeling and simulation of scientific workflows are needed to identify the interaction of workflows with programming models and systems for optimization.

We need to investigate if workflow tools might provide an opportunity for optimization at exascale. For example, can workflow systems enable automated data triage or reduction?; Can system interaction lead to auto-tuning?

Acknowledgements

[This is just cruft required to release a document written by a Sandia employee. It should be moved to where it makes sense.]

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.